CENG567: Homework #1

Yiğit Sever

November 8, 2020

1 Stable Matching

Question (a)

Use *Gale-Shapley* algorithm to find a stable matching for the following set of four colleges, four students and their preference lists.

Gale-Shapley algorithm, from lecture slides, edited for the context

```
Initialize each person to be free.
while (some student is free and hasn't applied to every college) {
    Choose such a student m
    c = 1st college on m's list to whom m has not yet applied
    if (c is free)
        assign c to m for potential application (a)
    else if (c prefers m to their current applicant m')
        assign m and c for potential application, and m' to be free (b)
    else
        c rejects m (c)
}
```

A quick trace of the algorithm;

1. S_1 is free;
   (a) applies to first college on their preference list C_4;
   (b) C_4 is free so it accepts and is matched with S_1 (a).
2. S_2 is free
   (a) applies to first college on their preference list; C_1
   (b) C_1 is free so it accepts and is matched with S_2 (a).
3. S_3 is free;
   (a) applies to first college on their preference list; C_1
   (b) C_1 rejects S_3 because it prefers S_2 to S_3 (c).
   (c) applies to second college on their preference list; C_2
   (d) C_2 is free so it accepts and is matched with S_3 (a).
4. S_4 is free;
   (a) applies to first college on their preference list; C_4

(b) C_4 rejects S_4 because it prefers S_1 to S_4 (c).

(c) applies to second college on their preference list; C_3

(d) C_3 is free so it accepts and is matched with S_4 (a).

5. There are no more free students to match, algorithm terminates.

The final matching and the answer to Question 1(a) is;

$$S_1 \to C_4$$
$$S_2 \to C_1$$
$$S_3 \to C_2$$
$$S_4 \to C_3$$

Question (b)

Find another stable matching with the same algorithm.

All executions of *Gale-Shapley* yield the same stable matching (that is proposer-optimal) and cannot produce *another* stable matching like the question text asks for.

Question (c)

Consider a pair of man m and woman w where m has w at the top of his preference list and w has m at the top of her preference list. Does it always have to be the case that the pairing (m, w) exist in every possible stable matching? If it is true, give a short explanation. Otherwise, give a counterexample.

Proof by contradiction. Assume that in the resulting matching of *Gale-Shapley*, we have S', where m is matched with w' and w is matched with m'.

The definition of stable matching dictates that there is *no incentive to exchange*, yet in S' m can trade up to w since m prefers w to w' and w can trade up since w prefers m to m'.

S' could not have occurred since men propose in accordance to their preference list, which w is on top of for m and no other men that may propose to w can make w switch since they are not more preffered than m.

Question (d)

Give an instance of n colleges, n students, and their preference lists so that the Gale-Shapley algorithm requires only $O(n)$ iterations, and prove this fact.

For a proposer agnostic formation, arrange the preference lists of colleges and students as follows;

$$C_1 \to \{S_1, S_n, S_{n-1}, \ldots, S_2\}$$
$$C_2 \to \{S_2, S_n, S_{n-1}, \ldots, S_3\}$$
$$\ldots$$
$$C_k \to \{S_k, S_n, S_{n-1}, \ldots, S_{k+1}\}$$
$$\ldots$$
$$C_n \to \{S_n, S_{n-1}, S_{n-2}, \ldots, S_1\}$$

$$
\begin{aligned}
S_1 &\to \{C_1, C_n, C_{n-1}, \ldots, C_2\} \\
S_2 &\to \{C_2, C_n, C_{n-1}, \ldots, C_3\} \\
&\ldots \\
S_k &\to \{C_k, C_n, C_{n-1}, \ldots, C_{k+1}\} \\
&\ldots \\
S_n &\to \{C_n, C_{n-1}, C_{n-2}, \ldots, C_1\}
\end{aligned}
$$

Where the student list $\{S_1, S_2, \ldots, S_n\}$ and college list $\{C_1, C_2, \ldots, C_n\}$ are shifted.

In this setup, ever proposer will propose to the first suitor in their preference list which is guaranteed to be free since they are not the first on any other suitor's preference list.

The algorithm in this instance runs in $\mathcal{O}(n)$ iterations, every proposer will follow the (a) branch in the algorithm given under Question 1.

Question (e)

Give another instance for which the algorithm requires $\Omega(n^2)$ iterations (that is, it requires at least cn^2 iterations for some constant $0 < c \leq 1$), and prove this fact.

Disclaimer: We have collaborated with CENG567 student Manolya Atalay for this question and used the answers and explanations given in the linked Mathematics Stack Exchange question[a].

[a] `https://math.stackexchange.com/questions/1410898/worst-case-for-the-stable-marriage-problem`

The worst problem instance for the algorithm (the instance that requires the largest number of steps) can be inferred as follows;

The highest number of times a man m can *propose* (every iteration has one proposal in it) to n many women is $n-1$. It cannot be n because it would mean that m did not find a suitable partner (rejected by everyone), which is contradictory with the algorithm's perfect matching guarantee.

For n men, in the worst case, each one can propose $n-1$ times; $n(n-1)$. One man has to propose one last time after getting rejected by $n-1$ women, giving the total number of proposals;

$$n(n-1) + 1 \tag{1}$$

As the theoretical highest number of iterations *Gale-Shapley* can have.

As for the instance that produces this run time;

Men have to arrange their preference list as follows for $n > 2$.

$$
M_k \to \begin{cases} W_k, W_{k+1}, \ldots, W_{n-1}, W_1, W_2, \ldots, W_n & \text{if } k \neq n \\ W_{k-1}, W_k, \ldots, W_{n-1}, W_1, W_2, \ldots, W_n & \text{if } k = n \end{cases} \tag{2}
$$

In other words, the *last* man's preference list is identical to one above them.

Women have to arrange their preference list as follows for $n > 2$;

$$
W_k \to \begin{cases} M_{k+1}, M_k, \ldots & \text{if } k \neq n-1 \\ M_1, M_n, \ldots & \text{if } k = n-1 \end{cases} \tag{3}
$$

The preference list of the last women W_n does not matter neither does the order of the rest of the men in the preference list of the women given.

We have implemented *Gale-Shapley* and an instance set creation script to test our findings;

n=3

```
n: Men Preference Table
A | 1 2 3
B | 2 1 3
C | 2 1 3
----
Women Preference Table
 1| B A C
 2| A C B
 3| A B C
----
Matching B with 2 (a)
Matching C with 2, B is now free (b)
Matching A with 1 (a)
Matching B with 1, A is now free (b)
Matching A with 2, C is now free (b)
C is rejected by 1 because B is more preffered (c)
Matching C with 3 (a)
Final pairings:
A - 2
B - 1
C - 3

A proposed 2 times
B proposed 2 times
C proposed 3 times
total of 7 times with n(n-1) + 1 = 7
```

n=6

```
n: Men Preference Table
A | 1 2 3 4 5 6
B | 2 3 4 5 1 6
C | 3 4 5 1 2 6
D | 4 5 1 2 3 6
E | 5 1 2 3 4 6
F | 5 1 2 3 4 6
----
Women Preference Table
 1| B A C D E F
 2| C B A D E F
 3| D C A B E F
 4| E D A B C F
 5| A F B C D E
 6| A B C D E F
----
Matching D with 4 (a)
Matching A with 1 (a)
Matching F with 5 (a)
E is rejected by 5 because F is more preffered (c)
Matching C with 3 (a)
Matching B with 2 (a)
// --- edited out for space ---
C is rejected by 1 because A is more preffered (c)
Matching C with 2, B is now free (b)
B is rejected by 3 because D is more preffered (c)
B is rejected by 4 because E is more preffered (c)
B is rejected by 5 because F is more preffered (c)
Matching B with 1, A is now free (b)
A is rejected by 2 because C is more preffered (c)
A is rejected by 3 because D is more preffered (c)
A is rejected by 4 because E is more preffered (c)
Matching A with 5, F is now free (b)
F is rejected by 1 because B is more preffered (c)
F is rejected by 2 because C is more preffered (c)
F is rejected by 3 because D is more preffered (c)
F is rejected by 4 because E is more preffered (c)
Matching F with 6 (a)
Final pairings:
E - 4
B - 1
A - 5
D - 3
C - 2
F - 6

A proposed 5 times
B proposed 5 times
C proposed 5 times
D proposed 5 times
E proposed 5 times
F proposed 6 times
total of 31 times with n(n-1) + 1 = 31
```

2 Stable Matching Variation

Question (a)

Consider a Stable Matching problem with men and women. Consider a woman w where she prefers man m to m', but both m and m' are low on her list of preferences. Can it be the case that by switching the order of m and m' on her list of preferences (i.e., by falsely claiming that she prefers m' to m) and running the algorithm with this modified preference list, w will end up with a man m'' that she prefers to both m and m'? Either give a proof that shows such an improvement is impossible, or give an example preference list for which an improvement for w is possible.

We will give an example instance. Let's take the example preference list for men; $M = x, y, z$ and women: $W = a, b, c$.

Table 1: Preference lists of women

Table 2: Everyone is being truthful

a	y	x	z
b	x	y	z
c	x	y	z

Table 3: a is lying about their preference

a	y	z	x
b	x	y	z
c	x	y	z

In the example given above in Table 1, we have two cases for women, Table 2 where everyone has given their actual preference list and Table 3 where a has lied about their 2^{nd} and 3^{rd} preferences.

The following is the men's preference table;

x	a	b	c
y	b	a	c
z	a	b	c

Table 4: The preference lists of the men.

The trace of *Gale-Shapley* algorithm for the truth telling case with Table 2 and Table 4 is below;

1. Matching x with a (a)
2. Matching y with b (a)
3. z proposes to a but gets rejected because a prefers x more (c)
4. z proposes to b but gets rejected because b prefers y more (c)
5. Matching z with c (a)

Which produces the following matching. Note that a is matched with their 2^{nd} choice.

$$x \to a$$
$$y \to b$$
$$z \to c$$

Now let's examine the trace of the algorithm when a lies by providing an altered preference table. The trace below uses Table 3 and Table 4;

1. Matching y with b (a)
2. Matching z with a (a)
3. **x proposes to a but gets rejected because a lies by saying that they like z more (c)**

4. Matching x with b, y is now free (b)
5. **Matching y with a, z is now free (b)**
6. z proposes to b but gets rejected because b prefers x more (c)
7. Matching z with c (a)

The highlighted lines show that a lies when proposed by x and then is able to trade up to y. The following it the final matching;

$$x \to b$$
$$y \to a$$
$$z \to c$$

Woman a was able to get their highest preferred option y by telling a lie. We have proved that an improvement is possible.

3 Asymptotics

Question (a)

What is the running time of this algorithm as a function of n? Specify a function f such that the running time of the algorithm is $\Theta(f(n))$.

Algorithm 1: Equivalent algorithm to one given in Question 3, edited for brevity

```
for i = 2; i < n; i+ = 1 do                    /* outer loop */
    for j = 1; j < n; j* = i do                /* inner loop */
        Some Θ(1) operation;
    end
end
```

The outer loop runs in linear time $\mathcal{O}(n)$ whereas the inner loop requires some deconstruction; Take the first iteration of the inner loop, $i = 2$ and j is initialized at 1.

$$\begin{aligned} 1^{\text{st}} \text{ iteration} &\to j = j * i \implies j = 2 \\ 2^{\text{nd}} \text{ iteration} &\to j = 4 \\ 3^{\text{rd}} \text{ iteration} &\to j = 8 \\ &\dots \\ m^{\text{th}} \text{ iteration} &\to j = i^m < n \end{aligned}$$

m is the last iteration of the inner loop because it hit the stopping condition $i^m < n$. Using the Equations above we can find the running time for the inner loop to hit stopping condition;

$$i^m < n$$
$$m < \log_i n$$

In other words, the inner loop has a running time in the order of $\mathcal{O}(\log n)$. By the product property of $\mathcal{O}$-notation we have the running time of $\mathcal{O}(n \log_a n), a > 1$ for the entire algorithm. The base a is useful to answer the rest of the question;

We can specify a function f such as $f = n\log_b(n)$ where $b > 1$. From the course slides;

$$\lim_{n\to\infty} \frac{n\log_a n}{n\log_b n} = c$$

And when the limit of two functions f, g converge to some constant c, then $f(n) = \Theta(g(n))$.

4 Big $\mathcal{O}$ and Ω

Question (a)

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove the following conjectures.

Q $f(n) = \mathcal{O}(g(n))$ implies $g(n) = \mathcal{O}(f(n))$

From the definition of Big $\mathcal{O}$ notation given in the lecture slides;

$$\exists\ c > 0 \quad \text{and} \quad n_0 \geq 0 \quad | \quad 0 \leq f(n) \leq c \cdot g(n) \quad \forall n \geq n_0 \tag{4}$$

If we rearrange the right hand side of the Proposition 4;

$$0 \leq \frac{1}{c} f(n) \leq g(n)$$

Or simply,

$$0 \leq c' f(n) \leq g(n) \tag{5}$$

By the definition of Big $\mathcal{O}$ notation, in order for $g(n) = \mathcal{O}(f(n))$ to be true,

$$\exists\ k > 0 \quad \text{and} \quad n_0' \geq 0 \quad | \quad 0 \leq g(n) \leq k \cdot f(n) \quad \forall n' \geq n_0'$$

Has to be true, yet from Equation 5 we know that $f(n)$ multiplied by some constant c' is strictly smaller than $g(n)$. The conjecture is *false*.

Q $f(n) = \mathcal{O}((f(n))^2)$

We can reuse the definition of Big $\mathcal{O}$ notation given in Equation 4, and using the question text, we have;

For $c > 1$ and $n_0 > 1$;

$$\begin{aligned} f(n) &\leq f(n)^2 \quad \forall n \geq n_0 \\ \text{take } g(n) &= f(n)^2 \\ f(n) &\leq c \cdot g(n) \quad \forall n \geq n_0 \end{aligned}$$

For $f(n) \geq 1$. The conjecture is true for asymptotically positive function $f(n)$ but does not hold for an $f(n) < 1$.

Q

$f(n) + o(f(n)) = \Theta(f(n))$

From the definition of Big Theta notation, we are trying to prove a relation such that;

$$0 \leq c_1 \cdot f(n) \leq f(n) + o(f(n)) \leq c_2 \cdot f(n) \tag{6}$$

First, let's give the little-o notation to remove $o(f(n))$ from Equation 6;

$$\forall\ c > 0 \quad \exists\ n_0 > 0 \mid\ 0 \leq g(n) < c \cdot f(n)\ \forall n \geq n_0 \tag{7}$$

So we can rewrite Equation 6 using Equation 7;

$$c_1 \cdot f(n) \leq f(n) + g(n) \leq c_2 \cdot f(n) \tag{8}$$

It's trivial to pick $c_1 < 1$ to deal with the left hand side of the inequality;

$$f(n) \leq f(n) + g(n)$$

For the right hand side; we can start by picking $n_0 = 1$ and $c = 1$ in the Equation 7; keeping $g(n)$ strictly smaller than $f(n)$.
In order to prove the right hand side of the inequality; we can pick $c_2 > 2$ which equals;

$$\begin{aligned} f(n) + g(n) &\leq 2 \cdot f(n) \\ \cancel{f(n)} + g(n) &\leq \cancel{2} \cdot f(n) \\ g(n) &\leq f(n) \end{aligned}$$

Which we proved above using Equation 7. This conjecture is *true*.

Question (b)

For each function $f(n)$ below, find (and prove that)

1. the smallest integer constant H such that $f(n) = \mathcal{O}(n^H)$
2. the largest positive real constant L such that $f(n) = \Omega(n^L)$

Otherwise, indicate that H or L do not exist.

Q

$f(n) = \frac{n(n+1)}{2}$

(1) For $n_0 \geq 1$; $\frac{n(n+1)}{2}$ is strictly smaller than n^2, yet $H = 1$ is not possible since through the definition of Big-$\mathcal{O}$ notation it implies that for a *constant* c;

$$\begin{aligned} 0 \leq \frac{n(n+1)}{2} &\leq c \cdot n \\ \frac{\cancel{n}(n+1)}{2} &\leq c \cdot \cancel{n} \\ \frac{n+1}{2} &\not\leq c \end{aligned}$$

Hence $f(n) = \mathcal{O}(n^2)$ and $H = 2$.
(2) $L = 2$. We can pick the constant $c = \frac{1}{2}$ and write the Big-Ω notation;

$$
\begin{aligned}
&c > 0 \text{ and } n_0 \geq 0 \\
&\frac{n(n+1)}{2} \geq \frac{1}{2}n^2 \\
&\frac{\cancel{n}(n+1)}{\cancel{2}} \geq \frac{1}{\cancel{2}}n^{\cancel{2}} \\
&n+1 \geq n
\end{aligned}
$$

Does a $L > 2$ work? Pick any $\ell > 2$;

$$
\begin{aligned}
&c > 0 \text{ and } n_0 \geq 0 \\
&\frac{n(n+1)}{2} \geq \frac{1}{2}n^{\ell} \\
&\frac{n(n+1)}{\cancel{2}} \geq \frac{1}{\cancel{2}}n^{\ell} \\
&n(n+1) \geq n^{\ell} \\
&n+1 \geq n^{\ell-1} \\
&1 \geq n(n^{\ell-2}-1)
\end{aligned}
$$

The final equation does not hold for $n > 1$ or the assumed $\ell > 2$ so $L > 2$ cannot be true.

Q $f(n) = \sum_{k=0}^{\lceil \log n \rceil} \frac{n}{2^k}$

It's beneficial to unpack the infinite sum beforehand;

$$
\begin{aligned}
f(n) &= \sum_{k=0}^{\lceil \log n \rceil} \frac{n}{2^k} \\
&= \frac{n}{2^0} + \frac{n}{2^1} + \frac{n}{2^2} + \cdots + \frac{n}{2^{\lceil \log n \rceil}} \\
&= n\Big(1 + \frac{1}{2^1} + \frac{1}{2^2} + \cdots + \frac{1}{2^{\lceil \log n \rceil}}\Big)
\end{aligned}
$$

The infinite sum approaches 1 as $\lceil \log n \rceil \to \infty$, giving us;

$$f(n) = 2 \cdot n$$

(1,2) Finally, $H = L = 1$ since $f(n)$ is $\Theta(n)$. Borrowing the Big-Θ notation from Equation 6 for $c_1 = 1$ and $c_2 = 2$;

$$n \leq f(n) \leq 2n$$

Q $f(n) = n(\log n)^2$

$$
\begin{aligned}
n^2 \quad &v.s. \quad n(\log n)^2 \\
n^1 \quad &v.s. \quad (\log n)^2 \\
\sqrt{n} \quad &v.s. \quad \sqrt{(\log n)^2} \\
n^{\frac{1}{2}} \quad &v.s. \quad \log n
\end{aligned}
$$

(1) From the lectures we know that $\log n$ is $\mathcal{O}(n^d)$ for all d.
If $\log n$ is $\mathcal{O}(n^{\frac{1}{2}})$ then $n(\log n)^2$ is $\mathcal{O}(n^2)$. $H = 2$.
(2) For L; we know that $f(n) = n(\log n)^2 > n^1$ but $n(\log n)^2 < n^2$ hence $L = 1$.